Contract-based automated integration for complex component-based systems

(Demo)

Johannes Schlatow and Rolf Ernst
Institute of Computer and Network Engineering, TU Braunschweig
{schlatow,ernst}@ida.ing.tu-bs.de

Marcus Nolte and Markus Maurer
Institute of Control Engineering, TU Braunschweig
{nolte,maurer}@ifr.ing.tu-bs.de

***Abstract*—In the scope of the research unit Controlling Concurrent Change, we developed a contract-based middleware to autonomously manage and ensure the safety, availability and security properties of a component-based run-time environment. It guarantees that any change to the system is formally analysed beforehand and only applied if it does not violate any of the contracts, thereby enabling in-field updateability of complex critical systems. For this purpose, a Multi-Change Controller (MCC) aggregates component contracts and invokes viewpoint-specific analysis engines to evaluate change requests and find feasible system configurations. The MCC is specifically designed for extensibility so that analysis engines can be added and combined dependent on the application domain. We present a demonstrator that showcases and illustrates this contract-based process for an automated integration of an automotive system. Our demonstrator is built upon the Genode OS Framework and Xilinx Zynq-7000 SoCs.**

I. Introduction

We enable the in-field updateability of critical component-based systems by using a contract-based approach. A component's contract formally specifies provided guarantees and assumptions that must be satisfied. Based on this, the viewpoint-specific analysis engines check configuration candidates and verify their feasibility before a configuration can be applied.

For this purpose, the software architecture is separated into an execution domain and a model domain. While the former executes the actual (critical) applications, the latter implements the on-demand contract-based integration process. A change request, such as a software update, triggers the Multi-Change Controller (MCC) to search a new configuration that is feasible w.r.t. all regarded viewpoints. A viewpoint is represented by its analysis engine, which evaluates the relevant parts of the contracts e.g. by applying existing models and established analyses. The MCC conducts a design space exploration assisted by a constraint solver (SAT or SMT) for which it iteratively invokes the analysis engines [1]. The MCC comprises different analysis engines, which implement the viewpoints relevant for the particular application domain.

II. Demonstrator

Our prototyping hardware platform is based on the Xilinx Zynq-7000 SoC, which comprises a processing system (ARM Cortex-A9) and an FPGA. This platform is particularly suited for computing-intensive environmental perception for which a growing interest can be observed in the automotive domain.

Our run-time environment is based on the Genode OS Framework [2], in which the application components are restricted to use service-oriented communication interfaces. This framework also acts as an abstraction layer to the underlying (micro-)kernel, which can therefore be exchanged during compile time in order to suit the requirements imposed by the application domain.

In the scope of this demo, we mainly focus on the viewpoints that solve the functional dependencies between the components as well as the mapping constraints that arise in a distributed system scenario. Hence, the MCC is obliged to find valid compositions of components and communication channels w.r.t. the service requirements and mapping constraints.

For this demo, we expose the state of the MCC to a host computer that allows us to inspect and visualise the component contracts, the viewpoint-specific models as well as the resulting configurations. This allows for an easy experimental evaluation of how this system reacts to incremental changes.

We illustrate the contract-based integration process by demonstrating two automotive showcases that have been developed for this purpose in the scope of the CCC project [3]. On the one hand, a cruise control showcase demonstrates how the functionality of an existing application can be incrementally extended depending on the available peripherals and software components. On the other hand, a park assistant showcase reveals the aspects that are additionally covered when different concurrent applications shall be integrated into the same platform.

Acknowledgements

This work was supported by the DFG Research Unit Controlling Concurrent Change (CCC), funding number FOR 1800. We thank the members of CCC for their support.

References

[1] J. Schlatow, M. Moestl, and R. Ernst, "An extensible autonomous reconfiguration framework for complex component-based embedded systems," in *12th International Conference on Autonomic Computing (ICAC 2015)*, Grenoble, France, July 2015, pp. 239–242.

[2] Genode OS Framework. [Online]. Available: http://genode.org/

[3] A. Reschka, M. Nolte, T. Stolte, J. Schlatow, R. Ernst, and M. Maurer, "Specifying a middleware for distributed embedded vehicle control systems," in *Proceedings of the 2014 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Hyderabad, India, December 2014, pp. 117–122.